SPECIFICATION

TO ALL WHOM IT MAY CONCERN:

BE IT KNOWN THAT I, Adam Seligman, have invented new and useful improvements in a

METHOD AND APPARATUS FOR DISPLAYING DISTRIBUTED MULTIRESOLUTION SCENES

of which the following is a specification:

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates generally to the storage and display of computer graphics. Specifically, the present invention manipulates and displays large sets of three-dimensional data using several discrete computer systems which operate together.

2. Description of the Related Art:

Computers are often used for the manipulation and display of graphical data, and computer users expect increasing levels of sophistication and realism in these systems. Interactive graphics have often been limited to two dimensional (2D) images and animated sequences, because the storage and computational requirements may be met with a typical computer system. Now, however, interaction with three dimensional (3D) scenes is utilized in many applications. These 3D scenes are often more difficult to manage than 2D graphics because of the storage and computational requirements necessary to properly view a 3D scene. Factors such as geometry and lighting calculations, applications of textures and other surface features must all be analyzed to accurately depict the 3D scene. Both the computation and storage of these elements are demanding of computer systems. However, due to advances in computer technology, modest amounts of 3D data can be rendered into images and displayed using generally available personal computers. But, very high-quality 3D scenes can require the storage and manipulation of more 3D data than a single personal computer can efficiently handle. Thus, the development of computer systems which can manipulate and display ever larger amounts of 3D data for generating high-quality graphic images is ongoing.

One solution has been to improve the hardware and software that converts 3D scene data into a sequence of images for display. However, the "3D rendering algorithms" used to accomplish this task are relatively advanced and only modest improvements have occurred in the past few years. The integrated circuits used for rendering have made significant advances, along with the progress of integrated circuits in general, but each new generation yields only incremental performance increases.

Another approach to improve the performance of 3D rendering algorithms relates to displaying only a portion of the 3D data at one time. By omitting large parts of data from the rendering process, the algorithm may execute more quickly. However, systems which rely solely on this method often are forced to omit relevant parts of the 3D scene. In essence, a "close up" view of one portion of the 3D scene is all that can be displayed. This characteristic is often unacceptable for users.

Multiresolution modeling represents yet another approach for efficiently manipulating and displaying 3D data sets. Multiresolution modeling techniques operate on the principle that some elements that make up a 3D scene are of less importance than others. For example, items in the background of a 3D scene might be represented in less detail than items in the foreground. These techniques correspond to the way people view the world around them. People perceive objects located far away as being generally smaller and containing less detail as compared to nearby objects. Computer systems employing multiresolution modeling techniques generally take advantage of this characteristic of human perception by illustrating objects located in the background of a 3D scene with less detail than those located in the foreground.

For example, a high-resolution image of a house could contain data which represents the contours of each brick, the leaves of shrubs in front of the house, the individual shingles on the roof, and so on. In a 3D scene, if the house is located in the immediate foreground, all of the data representing the details described above may have to be recalled from some storage device (*e.g.*, a hard disk or CD-ROM), transmitted to the graphics processor, and then rendered. On the other hand, if the house were in the far background, the bricks could be represented as simple rectangles of a single color, without any contours, and the shrubs and roof could be similarly simplified. The amount of data required to represent a low-resolution 3D model of the house is much less than is required to represent the high-resolution 3D model. Processing lower resolution 3D data results in faster rendering of the data into images, as the smaller data set size allows for a lightened computational load for the computer system.

Multiresolution modeling appears to be an efficient method for lessening the data sizes used to render, however, it does have its drawbacks. These drawbacks tend to adversely impact either disk storage space or processing resources. Figure 1 illustrates an example of how implementing multiresolution modeling on a computer system can consume large amounts of the computer's data storage (*e.g.*, disk space or RAM). As shown in Figure 1, several versions of 3D data representing the house discussed above are stored at varying levels of detail (LOD). The model designated as LOD[0] contains all of the details of the house, while model LOD[8] is a highly simplified model, with few detail. Models LOD[1] through LOD[7] contain intermediate levels of detail. When the 3D model of the house is required for the rendering of an image, workstation 120 determines the appropriate detail needed for the rendered house to have good visual qualities, then recalls the data representing the house at the appropriate LOD from storage 110. Next, workstation 120 uses this LOD model of the house in its graphics subsystem, which renders it to display 130. The problem with this method is that multiple data sets representing the same house at varying LOD have to be stored in storage 110. These redundant models consume more storage space than a single model and therefore represent an inefficient aspect of

this implementation of the multiresolution modeling technique. The storage penalty is particularly large if many LOD versions are stored for higher visual quality.

Some more sophisticated multiresolution imaging techniques use what are known as progressive techniques. Progressive techniques allow a single high-resolution 3D data set to be transformed into other data sets with an almost unlimited LOD. In particular, Hughues Hoppes' Progressive Mesh (PM) represents an advanced multiresolution progressive technique. In 3D graphics, vertices in 3D spaces describe polygons. Polygons are the basic building blocks in traditional 3D graphics. PM techniques generally operate on the collection of vertices that describe the surfaces of objects.

Using PM techniques, a 3D data set representing a complex object is converted into a simplified form made of many fewer vertices and a list (or stream) of "mesh operations" that can extend this data back into the complete 3D data set. The more a computer system utilizes the stream, the more details of the object represented by the 3D data will be available. A short prefix of the stream generates a low LOD version of the object, while a longer prefix of the stream generates a higher LOD version of the object. One benefit of PM techniques is that redundant copies of the model do not have to be stored at varying LOD, as the system can generate an LOD model needed dynamically. However, PM techniques generally require a significant amount of processing resources to convert a model from one LOD to another.

Thus, it would be desirable to create a computer system which uses PM techniques for high visual quality given the constraints of available hardware. However, such a system should not suffer excessive performance penalties associated with constructing lower-resolution data sets from the high-resolution sets, or the benefit of multiresolution modeling is lost.

SUMMARY OF THE INVENTION

This invention is a method of utilizing a collection of computers ("pool") to both store the 3D scene and assist in multiresolution processing. In one embodiment of the present invention, a visualization console acquires a 3D data set representing a 3D scene. The visualization console then transfers the 3D objects within the scene to a pool of other computers. These other computers associate identifiers with the 3D objects and store the 3D data across the pool of machines. Sometime later, the visualization console sends a request to the pool of storage computers for the data needed to render a view of the scene. After receiving the request, the pool of storage computers generates the 3D model with the appropriate LOD for each object, as specified by the request. The pool of storage computers then sends the appropriate LOD representation of the 3D objects to the visualization console for display in the requested scene. The visualization console hence does not need to necessarily perform any multiresolution processing, freeing its computation resources for rendering or other processes.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a prior art computer graphics system;

Figure 2 depicts a computer upon which the present invention can be implemented;

Figure 3 depicts a network of computers upon which the present invention can be implemented; and

Figure 4 is a flowchart illustrating a method of displaying a 3D scene according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Figure 2 depicts data processing system 300, which includes processor 302 and display 304. Display 304 includes display screen 306, which may be implemented utilizing a cathode ray tube (CRT), a liquid crystal display (LCD), a thin film transistor (TFT) panel, or the like. Data can be entered into data processing system 300 by means of a mouse 318 or keyboard 316. In addition to mouse 318 and keyboard 316, data can be entered using a track ball, joystick, touch sensitive tablet or screen, trackpad, or glidepad. Mouse 318 may be utilized to move a pointer or cursor on display screen 306. Processor 302 may also be coupled to one or more peripheral devices, such as modem 308 or disk drive 310, each of which may be internal or external to the enclosure of processor 302. Data processing system 300 may also be connected to network 312 in order to communicate with other computer units. Network 312 may be a local intranet (*e.g.,* a LAN or WAN), or the network of computers known as the Internet, or both. An output device such a printer 314 may also be coupled to processor 302.

Those persons skilled in the art of computer system design should recognize that display 304, keyboard 316, and the pointing device 318 may each be implemented utilizing any one of several known off-the-shelf components. Data processing system 300 may be implemented utilizing any general purpose computer or so-called "personal computer," "workstation," or "server," such as those sold by Compaq and others.

Figure 3 illustrates a computer network according to the present invention. Visualization console 404 is connected to workstations 402 by communication links 406. Executing on visualization console 404 is a 3D application program 410. This program provides a user interface and directs the operation of the graphics subsystem on visualization console 404. Also, the 3D application program manages information related to the 3D scenes that can be viewed on visualization console 404. Generally, the information for a given scene is contained in a data structure known as a scene representation 412. In the present invention, these scene representations 412 contain identifiers 414 for each object in a 3D scene, as well as the location of each object in the scene. In prior art implementations, the scene representations contained the 3D geometry and other data that described all the objects in the scene. However, in the present invention, preferably only identifiers 414 and locations for the objects are contained in the scene representation. The bulk of the information representing the objects is stored, at least in part, on workstations 402. The 3D application program uses the identifiers 414 as references to the objects 422 located on workstations 402. In a preferred embodiment of the present invention, the information representing the objects 422 will be stored in database structures on workstations 402, and the identifiers 414 will be used as indexes into these databases.

Operating on each workstation 402 is a scene database program 420. The scene database program 420 stores the 3D and other data representing the objects 422 in a given scene and processes this information upon receiving instructions from visualization console 404. Each workstation 402 can store the 3D data for an entire scene, or the scene can be divided between the various workstations in order to facilitate parallel processing of the scene.

To display an image on visualization console 404, a request is sent to workstations 402. This request contains the identifier 414 of the objects to be displayed. The scene database programs 420 operating on workstations 402 receive the request, then each of workstations 402 begins to construct the objects 422 specified in the requests. This construction process consists, in part, of retrieving the object 422 specified by the identifier 414 contained in the request from a storage medium (*e.g.*, a disk or RAM) accessible by the workstation, and performing some multiresolution and other 3D processing on the object. After workstations 402 have retrieved and processed the specified objects 422, the appropriate LOD models and other data representing the 3D scene are sent to visualization console 404 to be displayed.

In a preferred embodiment of the present invention, workstations 402 are rack mounted Compaq Professional Workstations with 1, 2, or 4, 600Mz processors, greater than 2 GB of RAM and large hard drives. Visualization console 404 is also a Compaq Professional Workstation which has a high performance rendering pipeline on a 3D OpenGL graphics card as its graphics subsystem. Connecting all of these computers is a Tandem ServerNet, which is a system area network (SAN) that communicates data between the various computer systems at the rate of 1 Gbit/s and which has a low communication latency.

Not shown in Figure 3 are the various network components such as network interface cards, bridges, routers, and the like, as the inclusion of these devices is understood by one of ordinary skill in the art. Further, the specific arrangement of computer devices mentioned above is but an exemplary embodiment. Other arrangements of computer devices can implement the invention described herein, and these alternative arrangements are within the scope of this invention.

Figure 4 is a flowchart illustrating a method for using the network of computers described in Figure 3 to display a rendered image from a 3D data set. For the sake of clarity, the following discussion will refer to only a single object 422. However, any number of objects up to the limits of the hardware and software, can be loaded, stored, processed, and displayed using the same method. To initialize a visualization console and workstation system, 3D data representing a 3D scene is acquired by the visualization console (step 502). The objects 422 which comprise the scene are then distributed to the workstations, along with their associated identifiers 414 (step 504). Information concerning the geometry, color, and texture of the object 422 is included with

the other information about each object. The visualization console works with the pool of workstations to distribute the objects 422 evenly among the pool of workstations, so when the 3D objects are processed, each workstation has approximately the same processing load. The purpose of this distribution is to have each workstation working in parallel with the others. A simple method of distribution is to randomly distribute the objects 422 to the workstations. Heuristics can also be used to distribute the objects 422 between the workstations.

When the objects 422 are received by the workstations, they are inserted into a database and associated with the identifier 414 that was received with the object (step 506). The objects 422 will usually be stored to disk, however, they can also be stored in the workstation's RAM. Objects stored in RAM can be retrieved much faster than objects stored to disk due to the respective access times between the two media.

Sometime later, when a user wishes to view a 3D scene that has been loaded into the workstations in the manner described above, the visualization console sends requests to the workstations to render the 3D scene (508). Normally, the request is formed in response to a user's input that specifies which portion of a given 3D scene the user wishes to view. In one embodiment, the visualization console determines which objects in the 3D scene need to be rendered, then directs the requests to the workstations which contain the objects. In an alternative embodiment, the visualization console can broadcast the parameters which describe the 3D scene the user wishes to view to the workstations. The workstations, instead of the visualization console, then determine which of the objects they manage need to be rendered. Information contained in the request include viewing parameters and lighting information.

Once a workstation receives a request from the visualization console, it uses the identifier 414 for the object to retrieve the representation of the object 422 from a database 420 accessible by the workstation. Once the information for an object 422 is retrieved from the database 420, the workstation will use the information transmitted in the request to create a PM with the appropriate LOD (step 510). During this processing, the workstation converts the PM mesh to a mesh with the specified LOD. This conversion includes computing the visibility and complexity of the object. The workstation then sends the specified LOD mesh representation of the object back to a rendering pipeline on visualization console (step 512). The LOD mesh sent to the visualization console by the workstation is a standard mesh of some resolution. In a preferred embodiment, the visualization console is unaware of the PM processing done by the workstations. The visualization console then displays the 3D data representing the objects 422 as it is received from the workstations (step 514).

In summary, the present invention uses several workstations to improve the speed at which large 3D scenes can be rendered. Full resolution models are converted to LOD models in

parallel on the workstations, thereby relieving the visualization console from the storage and computational task associated with the conversion process. Since PM techniques are utilized to simplify the 3D data, the 3D rendering pipeline has to process less data than would otherwise be possible. Operating in this manner allows the 3D rendering pipeline to display images faster while maintaining high visual quality. Also, the use of a high speed network allows the workstations and the visualization console to transfer data very quickly, thereby eliminating most of the performance penalties associated with distributing the processing of the objects.

Aspects of this invention pertain to specific "method functions" implementable on computer systems. In an alternate embodiment, the invention may be implemented as a computer program product for use with a computer system. Those skilled in the art should readily appreciate that programs defining the functions of the present invention can be delivered to a computer in many forms, which include, but are not limited to: (a) information permanently stored on non-writable storage media (e.g. read only memory devices within a computer such as ROMs or CD-ROM disks readable only by a computer I/O attachment); (b) information alterably stored on writable storage media (e.g. floppy disks and hard drives); or (c) information conveyed to a computer through communication media, such as a local area network, a telephone network, or a public network like the Internet. It should be understood, therefore, that such media, when carrying computer readable instructions that direct the method functions of the present invention, represent alternate embodiments of the present invention.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.